Available online at www.sciencedirect.com

SciVerse ScienceDirect

IERI Procedia 1 (2012) 59 – 65

Procedia IERI

www.elsevier.com/locate/procedia

2012 2nd International Conference on Mechanical, Industrial, and Manufacturing Engineering

# Self Adaptive Artificial Bee Colony for Global Numerical Optimization

Wenxiang Gu[a,b], Minghao Yin[a*], Chunying Wang[a]

[a]*College of Computer Science, Northeast Normal University, Changchun, 130117, P.R. China.*

[b]*Changchun Architecture & Civil Engineering College Department of Basic Subject Teaching, 130607, P.R. China.*

**Abstract**

The ABC algorithm has been used in many practical cases and has demonstrated good convergence rate. It produces the new solution according to the stochastic variance process. In this process, the magnitudes of the perturbation are important since it can affect the new solution. In this paper, we propose a self adaptive artificial bee colony, called self adaptive ABC, for the global numerical optimization. A new self adaptive perturbation is introduced in the basic ABC algorithm, in order to improve the convergence rates. 23 benchmark functions are employed in verifying the performance of self adaptive ABC. Experimental results indicate our approach is effective and efficient. Compared with other algorithms, self adaptive ABC performs better than, or at least comparable to the basic ABC algorithm and other state-of-the-art approaches from literature when considering the quality of the solution obtained.



Selection and peer review under responsibility of Information Engineering Research Institute

Keywords*: artificial bee colony; self adaptive; global numerical optimization; exploitation;*

* Corresponding author. Tel.: +8613039221514; fax: +86-0431-84536331.

*E-mail address:* ymh@nenu.edu.cn.

This work is supported by the National Natural Science Foundation of China (61070084).

 doi:10.1016/j.ieri.2012.06.011

## 1. Introduction

Optimization problems play an important role in both industrial application fields and the scientific research world. During the past decade, we have viewed different kinds of meta-heursitic algorithms advanced to handle optimization problems. Among them, Meta-heuristic based methods, such as simulated annealing (SA), genetic algorithm (GA), particle swarm optimization algorithm (PSO), artificial bee colony (ABC), and differential evolution [1-4], may be one of the most popular methods.

Particularly, artificial bee colony algorithm [5] is a population-based heuristic evolutionary algorithm inspired by the intelligent foraging behavior of the honeybee swarm. B. Akay and D. Karaboga [6] proposed a modify versions of the artificial bee colony. The modified artificial bee colony applied for efficiently solving real parameter optimization problem. The modified ABC algorithm employs a control parameter that determines how many parameters to be modified for the production of a neighboring solution. A scaling factor that tunes the step size adaptively was introduced. However, this field of study is still in its early days, a large number of future researches are necessary in order to develop the new version artificial bee colony algorithm for optimization problems.

Since ABC is a particular instance of EA, it is interesting to investigate how self adaptive can be applied to it. Until now, there is no paper to focus on self-adaptive in ABC has been reported. In our paper, the parameter control technique is based on the self adaptive of the magnitudes of the perturbation, associated with the evolutionary process. The main goal here produces a flexible ABC, in terms of control parameter. We propose a self adaptive ABC which the variant of control parameter is changed according to the iteration. The low value of control parameter allows the search to find the optimal search in small step. However, it makes the convergence slower. A high value of control parameter speed up the search but it reduces the exploitation capability of the perturbation process. Therefore, we use this method which can balance the exploration and the exploitation of ABC.

## 2. Self Adaptive ABC

Artificial Bee colony is an evolutionary algorithm first introduced by Karaboga in 2005. This algorithm simulates the foraging behavior of the bee colony. In this algorithm, the model of the ABC algorithm consists of three groups of bees: employed bees, onlooker bees, and scout bees. For each food source, there is only one employed bee. In other words, the number of bees is equal to the number of food sources. Employed bees are responsible for exploiting the nectar sources explored before, sharing their information with onlookers within the hive. After that, the onlookers will select one of the food sources within the neighborhood of the food source. An employed bee becomes a scout if the food source is abandoned, and then starts to search a new food source randomly [4].

In basic ABC, employed bees are responsible for exploiting the nectar sources explored before and giving information to the waiting bee in the hive about the quality of the food source sites which they are exploiting. Onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bees. Scouts either randomly search the environment in order to find a new food source depending on an internal motivate or based on possible external clues. The ABC produces the new solution according to the stochastic variance process. In this process, the magnitudes of the perturbation are important since it can affect the new solution. In order to improve the convergence rates, a new self adaptive perturbation will be introduced in the basic ABC algorithm.

The self adaptive artificial bee colony algorithm is proposed based on the structure of basic ABC algorithm. By employing the self adaptive iteration according the relative maximum iterations, the employed bee and onlooker bee can be exploited. In the basic ABC, a random perturbation which avoids getting stuck at local minima is added to the current solution in order to produce the new solution. This random perturbation

is the difference of the solution weighted by the $\phi_{ij}$. The $\phi_{ij} = (rand - 0.5) \times 2$ is a uniformly distributed real random number within the range $[-1,1]$ in the basic ABC. In the self adaptive ABC, the variant of $\phi_{ij}$ is changed according to the iteration. For the employed bee, the variant of $\phi_{ij}$ is used the equation (1). For the onlooker bee, the variant of $\phi_{ij}$ can use the equation (2).

For the employed bee colony:

$$\phi_{i,j} = \begin{cases} e^{-\frac{0.08 \times iter}{\max iter}} & \text{rand} < 0.5 \\ -e^{-\frac{0.08 \times iter}{\max iter}} & \text{otherwise} \end{cases} \quad (1)$$

For Onlooker bee colony

$$\phi_{i,j} = \begin{cases} e^{-\frac{4 \times iter}{\max iter}} & \text{rand} < 0.5 \\ -e^{-\frac{4 \times iter}{\max iter}} & \text{otherwise} \end{cases} \quad (2)$$

The low value of $\phi_{ij}$ allows the search to find the optimal search in small step. However, it makes the convergence slower. A high value of $\phi_{ij}$ speed up the search but it reduces the exploitation capability of the perturbation process. Therefore, we use this method which can balance the exploration and the exploitation of ABC. The self adaptive ABC has a very simple structure and thus is easy to implement. This method can overcome the lack of the exploitation and exploration of the basic algorithm.

The main contribution of our approach is that user does not use the rand $\phi_{ij}$. The variant of $\phi_{ij}$ can change according to the iteration. The rule of the self adaptive ABC is quite simple; therefore the new version of the ABC algorithm does not increase the time complexity, in comparison to the original DE algorithm.

## 3. Experimental results

### *3.1 Experimental setup*

To evaluate the performance of our algorithm, we applied it to 23 standards benchmark functions. These functions have been widely used in the literature. Since we do no make any modification of these functions, they are given in the paper [7]. In order to evaluate the effectiveness and efficiency of self adaptive ABC, we have chosen a suitable set of value and have not made any effort in finding the best parameter settings. In this experiment, we set the number of Food source to be 100, set limit to be 100. In this strategy, all vectors for mutation are selected from the population at random and, then, it has no bias to any special search directions and chooses new search directions in a random manner. The maximum number of generations: 1500 for *f1,f6,f10,f12, and f13, 2000 for f2 and f11, 3000 for f7,f8,f9, and 4000 for f15,5000 for f3,f4,f5. 100 for f14,f16-f19,f21,f22,f23.* For all test functions, the algorithms carry out 50 independent runs.

### *3.2 Comparison of self adaptive ABC, ABC, DE*

In order to show the effectiveness of our proposed self adaptive ABC approach, we compare it with the original ABC algorithm and the DE algorithm. Differential Evolution (DE) [3] is an Evolutionary Algorithm first introduced by Storn and Price. Similar to other evolutionary algorithms particularly genetic algorithm,

DE uses some evolutionary operators like selection recombination and mutation operators. Different from genetic algorithm, DE use distance and direction information from current population to guide the search process. The crucial idea behind DE is a scheme for producing trial vectors according to the manipulation of target vector and difference vector .If the trail vector yields a lower fitness than a predetermined population member, the newly trail vector will be accepted and be compared in the following generation. In the experiment, the mean results of 50 independent runs are summarized in Table 1. Compared with the DE algorithm, from Table 1 we can see that the self adaptive ABC is significantly better than DE on 10 functions. However, DE is outperformed by self adaptive ABC on three functions (*f03, f04, f07*). For the rest 10 functions, there are no significant differences. For the multimodal function with many local minimum, i.e. *f08-f13*, it is clear that the best results are obtained by self adaptive ABC. DE may trap into the local minima for three out of six functions. The self adaptive ABC can find better solutions than DE algorithm within the Max_NFFEs. This result illustrates the algorithm has better ability to escape from poor local optima and locate a good near-global optimum.

Compared with ABC: From Table 2, it is obviously that self adaptive ABC performs better solutions than ABC for four unimodal functions. For *f04* and *f05*, ABC is better than self adaptive ABC. For multimodal function *f8, f10, f12, f13*, the self adaptive ABC can provide better solutions than ABC, For *f09* and *f11*, there are no significant differences. For *f14,f 16,f18, f19* with only a few local minima, the dimension of the function is also small. In this case, it is hard to judge the performances of individual algorithms. All algorithms were able to find optimal solutions for these two functions. For *f15* and *f17*, the self adaptive ABC can find the better solution than ABC. For *f21 –f23*, there is no superior algorithm either. For *f21- f23,* ABC algorithm is better than self adaptive.

### *3.3 Comparison of Self adaptive ABC and ABC algorithm with FEP and CEP algorithms*

In the experiment, we compare the performance of self adaptive ABC and ABC with FEP and CEP. The average results of 50 independent runs are summarized in Table 2. Result for the FEP and CEP algorithms are taken from [7]. From Table 2, the comparison shows that self adaptive ABC gives better results on benchmark function than FEP and CEP. Self adaptive ABC is able to obtain smaller standard deviations of function values. For the unimodal function *f1*, *f2 and f6*, the Self adaptive ABC can give the better solution than FEP and CEP algorithm. For f03 and f04, FEP perform better than self adaptive ABC algorithm and CEP, For multimodal functions *f8-f13* with many local minima, The self adaptive ABC provided better solutions than FEP and CEP algorithm. For *f14, f16-f20*, there is no superior algorithm either. For *f15*, the self adaptive ABC provided better solutions than FEP and CEP algorithm. For *f21-f23*, self adaptive ABC and ABC algorithm performs better than FEP and CEP algorithm. It means that the solution quality of self adaptive ABC is more stable than FEP and CEP.

## 4. Conclusions

In this paper, we propose a self adaptive artificial bee colony, called self adaptive ABC. The proposed self adaptive method is an attempt to determine the values of control parameter $\phi_{ij}$ . Our self adaptive ABC algorithm has been implemented and test on benchmark optimization problems taken from literature. 23 benchmark functions chosen from literature are employed. The results show that the proposed self adaptive ABC algorithm clearly outperforms the basic ABC. Compared with some evolution algorithms from literature, we find our algorithm is superior to or at least highly competitive with these algorithms.

In this paper, we only consider the global optimization. The algorithm can be extended to solve other problem such as constrained optimization problems.

**Acknowledgements**

This research is fully supported by the National Natural Science Foundation of China under Grant Nos.60473042, 60573067, 60803102.

**References**

[1] Hsieh, S.T., Sun, T.Y., Liu, C.C., "Potential offspring production strategies: An improved genetic algorithm for global numerical optimization". Expert Systems with Applications, vol.36, no.8, pp.11088-11098, 2009.
[2] Clerc, M., Kennedy, J., "the Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space." IEEE Transactions on Evolutionary Computation, vol.6, pp.58-73 ,2002.
[3] Storn, R. , Price, K., "Differential evolution-a simple and efficient heuristic for global optimization over continuous space." Journal of Global Optimization,vol.11, pp.341-359, 1997.
[4] Karaboga, D., Basturk, B., "On The Performance Of Artificial Bee Colony (ABC) Algorithm." Applied Soft Computing, vol.8, no.1, pp. 687-697, 2008.
[5] Karaboga, D., Basturk, B., "A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm." Journal of Global Optimization, vol.39, no.3) , pp. 459-171, 2007.
[6] Akay, B., Karaboga D., "A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization." Information Sciences, doi:10.1016/j.ins.2010.07.015, 2011.
[7] Yao, X., Liu, Y., Lin, G., "Evolutionary programming made faster." IEEE Transactions on Evolutionary Computation, 3(2), 82–102, 1999.

Table 1 Comparisons of ABC,DE,Self adaptive ABC

| Function | Max_FES | DE | | ABC | | Self adaptive ABC | |
|---|---|---|---|---|---|---|---|
| | | Mean | std dev | Mean | std dev | Mean | std dev |
| *f01* | 150000 | 5.2833e-014 | 3.5135e-014 | 2.6354e-020 | 1.9081e-020 | 2.8991e-026 | 3.6025e-026 |
| *f02* | *200000* | 7.2007e-010 | 5.6668e-010 | 1.2031e-015 | 4.7318e-016 | 1.9991e-018 | 7.8506e-019 |
| *f03* | *500000* | 1.8283e-011 | 1.6653e-011 | 2.8030e+003 | 772.0086 | 2.1476e+003 | 1.3626e+003 |
| *f04* | *500000* | 0.0860 | 0.1075 | 2.8748 | 0.8807 | 8.4832 | 1.5395 |
| *f05* | *500000* | 0.2657 | 1.0293 | 0.0518 | 0.0393 | 0.1903 | 0.2792 |
| *f06* | *150000* | 0 | 0 | 0 | 0 | *0* | *0* |
| f07 | *300000* | 0.0044 | 7.6711e-004 | 0.1660 | 0.0190 | 0.1797 | 0.0375 |
| *f* 8 | 300000 | -1.25457e+004 | 66.3979 | -1.2569e+004 | 5.9466e-005 | -1.2569e+004 | 4.6559e-012 |
| *f* 9 | 300000 | 1.2869e+002 | 28.1729 | 1.2506e-013 | 8.1648e-014 | 8.1476e-014 | 3.8592e-014 |
| *f* 10 | 150000 | 2.2021e-008 | 6.0555e-009 | 1.0925e-009 | 5.4664e-010 | 1.2486e-012 | 3.7227e-013 |
| *f* 11 | 200000 | 4.9306e-004 | 0.0019 | 6.1617e-015 | 1.2743e-014 | 7.8763e-014 | 3.4904e-013 |
| *f* 12 | 150000 | 6.9083e-015 | 8.2614e-015 | 8.8643e-021 | 1.0071e-020 | 9.1663e-028 | 1.3011e-027 |
| *f* 13 | 150000 | 2.5765e-014 | 1.9767e-014 | 5.5921e-020 | 5.9975e-020 | 1.0737e-026 | 1.0685e-026 |
| *f* 14 | 10000 | 0.9980 | 2.7195e-016 | 0.9980 | 5.7500e-014 | 0.9980 | 5.8312e-017 |
| *f* 15 | 400000 | 4.5e-4 | 3.3e-4 | 6.9543e-004 | 1.5033e-004 | 4.8215e-004 | 1.4548e-004 |
| *f* 16 | 10000 | -1.0316 | 7.1858e-013 | -1.0316 | 2.1656e-014 | -1.0316 | 4.3441e-016 |
| *f* 17 | 10000 | 0.3979 | 0 | 0.3979 | 1.1700e-008 | 0.3979 | 0 |
| *f* 18 | 10000 | 3 | 2e-015 | 3.0000 | 2.8278e-009 | 3.0000 | 1.4768e-009 |
| *f* 19 | 10000 | -3.8628 | 9.4950e-016 | -3.8628 | 2.1202e-010 | -3.8628 | 3.9477e-014 |
| *f* 20 | 20000 | -3.2665 | 0.0614 | -3.3220 | 1.6925e-008 | -3.3220 | 2.2190e-010 |
| *f* 21 | 10000 | -10.1532 | 3.3577e-006 | -10.1516 | 0.0056 | -10.0526 | 0.3320 |
| *f* 22 | 10000 | -10.4029 | 9.8943e-006 | -10.4020 | 0.0028 | -10.0946 | 0.9429 |
| *f* 23 | 10000 | -10.5364 | 4.7510e-007 | -10.5334 | 0.0060 | -10.2698 | 0.7886 |

Table 2 Comparisons of, FEP, CEP, ABC, ,Self adaptive ABC

| F | Mean best | | | | std | | | |
|---|---|---|---|---|---|---|---|---|
| | FEP | CEP | ABC | Self adaptive ABC | FEP | CEP | ABC | Self adaptive ABC |
| *f01* | 5.7e-4 | 2.2e-4 | 2.6354e-020 | 2.8991e-026 | 1.3e-4 | 5.9e-4 | 1.9081e-020 | 3.6025e-026 |
| *f02* | 8.1e-3 | 2.6e-3 | 1.2031e-015 | 1.9991e-018 | 7.7e-4 | 1.7e-4 | 4.7318e-016 | 7.8506e-019 |
| *f03* | 1.6e-2 | 5.0e-2 | 2.8030e+003 | 2.1476e+003 | 1.4e-2 | 6.6e-2 | 772.0086 | 1.3626e+003 |
| *f04* | 0.3 | 2.0 | 2.8748 | 8.4832 | 0.5 | 1.2 | 0.8807 | 1.5395 |
| *f05* | 5.06 | 6.17 | 0.0518 | 0.1903 | 5.87 | 13.61 | 0.0393 | 0.2792 |
| *f06* | 0 | 577.76 | 0 | *0* | 0 | 1125.76 | 1.8241e-020 | *3.7565e-026* |
| *f07* | 7.6e-3 | 1.8e-3 | 0.1660 | 0.1797 | 2.6e-3 | 6.4e-3 | 0.0190 | 0.0375 |
| *f08* | -12554.5 | -7917.1 | -1.2569e+004 | -1.2569e+004 | 52.6 | 634.5 | 5.9466e-005 | 4.6559e-012 |
| *F09* | 4.6e-2 | 89.0 | 1.2506e-013 | 8.1476e-014 | 1.2e-2 | 23.1 | 8.1648e-014 | 3.8592e-014 |
| *f10* | 1.8e-2 | 9.2 | 1.0925e-009 | 1.2486e-012 | 2.1e-3 | 2.8 | 5.4664e-010 | 3.7227e-013 |
| *f11* | 1.6e-2 | 8.6e-2 | 6.1617e-015 | 7.8763e-014 | 2.2e-2 | 0.12 | 1.2743e-014 | 3.4904e-013 |
| *f12* | 9.2e-6 | 1.76 | 8.8643e-021 | 9.1663e-028 | 3.6e-6 | 2.4 | 1.0071e-020 | 1.3011e-027 |
| *f13* | 1.6e-4 | 1.4 | 5.5921e-020 | 1.0737e-026 | 7.3e-5 | 3.7 | 5.9975e-020 | 1.0685e-026 |
| *f14* | 1.22 | 1.66 | 0.9980 | 0.9980 | 0.56 | 1.19 | 5.7500e-014 | 5.8312e-017 |
| *f15* | 5.0e-4 | 4.7e-4 | 6.9543e-004 | 4.8215e-004 | 3.2e-4 | 3.0e-4 | 1.5033e-004 | 1.4548e-004 |
| *f16* | -1.03 | -1.03 | -1.0316 | -1.0316 | 4.9e-7 | 4.9e-7 | 2.1656e-014 | 4.3441e-016 |
| *f17* | 0.398 | 0.398 | 0.3979 | 0.3979 | 1.5e-7 | 1.5e-7 | 1.1700e-008 | 0 |
| *f18* | 3.02 | 3.0 | 3.0000 | 3.0000 | 0.11 | 0 | 2.8278e-009 | 1.4768e-009 |
| *f19* | -3.86 | -3.86 | -3.8628 | -3.8628 | 1.40e-5 | 1.40e-2 | 2.1202e-010 | 3.9477e-014 |
| *f20* | -3.27 | -3.28 | -3.3220 | -3.3220 | 5.9e-2 | 5.8e-2 | 1.6925e-008 | 2.2190e-010 |
| *f21* | -5.52 | -6.86 | -10.1516 | -10.0526 | 1.59 | 2.67 | 0.0056 | 0.3320 |
| *f22* | -5.52 | -8.27 | -10.4020 | -10.0946 | 2.12 | 2.95 | 0.0028 | 0.9429 |
| *f23* | -6.57 | -9.10 | -10.5334 | -10.2698 | 3.14 | 2.92 | 0.0060 | 0.7886 |